NAME

prep, fdisk, format, mbr – prepare disks, floppies and flashes

SYNOPSIS

disk/prep [–bcfnprw] [–a name]... [–s sectorsize] plan9partition

disk/fdisk [–abfprw] [–s sectorsize] disk

disk/format [–dfvx] [–b bootblock] [–c csize] [–l label] [–r nresrv] [–t type] disk [file...]

disk/mbr [–9] [–m mbrfile] disk

DESCRIPTION

A partition table is stored on a non–floppy disk to specify the division of the physical disk into a set of logical units. On PCs, the partition table is stored at the end of the master boot record of the disk. Partitions of type 0x39 are Plan 9 partitions. The names of PC partitions are chosen by convention from the type: dos, plan9, etc. Second and subsequent partitions of the same type on a given disk are given unique names by appending a number (or a period and a number if the name already ends in a number).

Plan 9 partitions (and Plan 9 disks on non–PCs) are themselves divided, using a textual partition table, called the Plan 9 partition table, in the second sector of the partition (the first is left for architecture–specific boot data, such as PC boot blocks). The table is a sequence of lines of the format part name start end, where start and end name the starting and ending sector. Sector 0 is the first sector of the Plan 9 partition or disk, regardless of its position in a larger disk. Partition extents do not contain the ending sector, so a partition from 0 to 5 and a partition from 5 to 10 do not overlap.

The Plan 9 partition often contains a number of conventionally named subpartitions. They include:

9fat A small FAT file system used to hold configuration information (such as plan9.ini and plan9.nvr) and kernels. This typically begins in the first sector of the partition, and contains the partition table as a ``reserved'' sector. See the discussion of the –r option to format. arenas A venti(8) arenas partition.

bloom A venti(8) bloom–filter partition.

cache A cfs(4) file system cache.

fossil A fossil(4) file system.

fs A kfs(4) file system.

fscfg A few–sector partition used to store an fs(3) configuration.

isect A venti(8) index section.

nvram A one–sector partition used to simulate non–volatile RAM on PCs.

other A non–archived fossil(4) file system.

swap A swap(8) swap partition.

fdisk and prep

Fdisk edits the PC partition table and is usually invoked with a disk like /dev/sdC0/data as its argument, while prep edits the Plan 9 partition table and is usually invoked with a disk partition like /dev/sdC0/plan9 as its argument. Fdisk works in units of disk ``cylinders'': the cylinder size in bytes is printed when fdisk starts. Prep works in units of disk sectors, which are almost always 512 bytes. Fdisk and prep share most of their options:

–a Automatically partition the disk. Fdisk will create a Plan 9 partition in the largest unused area on the disk, doing nothing if a Plan 9 partition already exists. If no other partition on the disk is marked active (i.e. marked as the boot partition), fdisk will mark the new partition active. Prep's –a flag takes the name of

a partition to create. (See the list above for partition names.) It can be repeated to specify a list of partitions to create. If the disk is currently unpartitioned, prep will create the named partitions on the disk, attempting to use the entire disk in a sensible manner. The partition names must be from the list given above.

–

b Start with a blank disk, ignoring any extant partition table.

–p Print a sequence of commands that when sent to the disk device's ctl file will bring the partition table information kept by the sd(3) driver up to date. Then exit. Prep will check to see if it is being called with a disk partition (rather than an entire disk) as its argument; if so, it will translate the printed sectors by

the partition's offset within the disk. Since fdisk operates on a table of unnamed partitions, it assigns names based on the partition type (e.g., plan9, dos, ntfs, linux, linuxswap) and resolves collisions by appending a numbered suffix. (e.g., dos, dos.1, dos.2).

–

r In the absence of the –p and –w flags, prep and fdisk enter an interactive partition editor; the –r flag runs the editor in read–only mode.

–s sectorsize

Specify the disk's sector size. In the absence of this flag, prep and fdisk look for a disk ctl file and read it to find the disk's sector size. If the ctl file cannot be found, a message is printed and a sector size of 512 bytes is assumed.

–

w Write the partition table to the disk and exit. This is useful when used in conjunction with –a or –b.

If neither the –p flag nor the –w flag is given, prep and fdisk enter an interactive partition editor that operates on named partitions. The PC partition table distinguishes between primary partitions, which can be listed in the boot sector at the beginning of the disk, and secondary (or extended) partitions, arbitrarily many of which may be chained together in place of a primary partition. Primary partitions are named pn, secondary partitions sn. The number of primary partitions plus number of contiguous chains of secondary partitions cannot exceed four.

The commands are as follows. In the descriptions, read ``sector'' as ``cylinder'' when using fdisk.

a name [start [end]]

Create a partition named name starting at sector offset start and ending at offset end. The new partition will not be created if it overlaps an extant partition. If

start or end are omitted, prep and fdisk will prompt for them. In fdisk, the newly created partition has type ``PLAN9;'' to set a different type, use the t command (q.v.). Start and end may be expressions using the operators +, –, *, and /, numeric constants, and the pseudovariables . and $. At the start of the program, . is set to zero; each time a partition is created, it is set to the end sector of the new partition. It can also be explicitly set using the . command. When evaluating start, $ is set to one past the last disk sector. When evaluating end, $ is set to the maximum value that end can take on without running off the disk or into another partition. Numeric constants followed by k, m, g, or t (or upper–case equivalents) are scaled to the respective size in kilo–, mega–, giga–, or tera–bytes. Finally, the expression n% evaluates to (nxdisksize)/100. As examples, a . .+20% creates a new partition starting at . that takes up a fifth of the disk, a . .+21G creates a new partition starting at . that takes up 21 gigabytes (21x230 bytes), and a 1000 $ creates a new partition starting at sector 1000 and extending as far as possible.

. newdotSet the value of the variable . to newdot, which is an arithmetic expression as described in the discussion of the a command.

d name Delete the named partition.

h Print a help message listing command synopses.

p Print the disk partition table. Unpartitioned regions are also listed. The table consists of a number of lines containing partition name, beginning and ending sectors, and total size. A ' is prefixed to the names of partitions whose entries have been modified but not written to disk. Fdisk adds to the end of

each line a textual partition type, and places a * next to the name of the active partition (see the A command below).

P Print the partition table in the format accepted by the disk's ctl file, which is also the format of the output of the –p option.

w Write the partition table to disk. Prep will also inform the kernel of the changed partition table. The write will fail if any programs have any of the disk's partitions open. If the write fails (for this or any other reason), prep and fdisk will attempt to restore the partition table to its former state. q Quit the program. If the partition table has been modified but not written, a warning is printed. Typing q again will quit the program.

Fdisk also has the following commands.

A name Set the named partition active. The active partition is the one whose boot

block is used when booting a PC from disk.

e Print the names of empty slots in the partition table, i.e., the valid names to use when creating a new partition.

t [type]Set the partition type. If it is not given, fdisk will display a list of choices and then prompt for it.

format and pbs

Format prepares for use the disk partition or the floppy diskette in the file named disk, for example /dev/sdC0/9fat or /dev/fd0disk. The options are:

–f Do not physically format the disc. Used to install a FAT file system on a previously formatted disc. If disk is not a floppy device, this flag is a no–op.

–t specify a density and type of disk to be prepared. The possible types are:

3½DD3½" double density, 737280 bytes

3½HD3½" high density, 1474560 bytes

5¼DD5¼" double density, 368640 bytes

5¼HD5¼" high density, 1146880 bytes

hardfixed disk

The default when disk is a floppy drive is the highest possible on the device. When disk is a regular file, the default is 3½HD. When disk is an sd(3) device, the default is hard.

–

d initialize a FAT file system on the disk.

–b use the contents of bootblock as a bootstrap block to be installed in sector 0.

The remaining options have effect only when –d is specified:

–c use a FAT cluster size of csize sectors when creating the FAT.

–l add a label when creating the FAT file system.

–r mark the first nresrv sectors of the partition as ``reserved''. Since the first

sector always contains the FAT parameter block, this really marks the nresrv–1 sectors starting at sector 1 as ``reserved''. When formatting the 9fat partition, –r 2 should be used to jump over the partition table sector.

Again under –d, any files listed are added, in order, to the root directory of the FAT file system. The files are contiguously allocated. If a file is named 9load, it will be created with the SYSTEM attribute set so that dossrv(4) keeps it contiguous when modifying it.

Format checks for a number of common mistakes; in particular, it will refuse to format a 9fat partition unless –r is specified with nresrv larger than two. It also refuses to format a raw sd(3) partition that begins at offset zero in the disk. (The beginning of the disk should contain an fdisk partition table with master boot record, not a FAT file system or boot block.) Both checks are disabled by the –x option. The –v option prints debugging information.

The file /386/pbs is an example of a suitable bootblock to make the disk a boot disk. It gets loaded by the BIOS at 0x7C00, reads the first sector of the root directory into address 0x7E00, and looks for a directory entry named 9LOAD. If it finds such an entry, it uses single sector reads to load the file into address 0x10000 and then jumps to the loaded file image. The file /386/pbslba is similar, but because it uses LBA addressing (not supported by older BIOSes), it can access more than the first 8.5GB of the disk. /386/pbsraw is suitable for CDs.

mbr

Mbr installs a new boot block in sector 0 (the master boot record) of a disk such as /dev/sdC0/data. If mbrfile contains more than one sector of `boot block', the rest will be copied into the first track of the disk, if it fits. This boot block should not be confused with the boot block used by format, which goes in sector 0 of a partition. Typically, the boot block in the master boot record scans the PC partition table to find an active partition and then executes the boot block for that partition. The partition boot block then loads a bootstrap program such as 9load (see 9boot(8)), which then loads the operating system. If MS–DOS or Windows is already installed on your disk, the master boot record already has a suitable boot block. Otherwise, /386/mbr is an appropriate mbrfile. It detects and uses LBA addressing when available from the BIOS (the same could not be done in the case of pbs due to space considerations). If the mbrfile is not specified, a boot block is installed that prints a message explaining that the disk is not bootable. The –9 option initialises

the partition table to consist of one plan9 partition which spans the entire disc starting at the end of the first track.

EXAMPLES

Initialize the kernel disk driver with the partition information from the FAT boot sectors. If Plan 9 partitions exist, pass that partition information as well.

```
for(disk in /dev/sd??) {
if(test –f $disk/data && test –f $disk/ctl)
disk/fdisk –p $disk/data >$disk/ctl
for(part in $disk/plan9*)
if(test –f $part)
disk/prep –p $part >$disk/ctl
}
```

Create a Plan 9 boot floppy on a previously formatted diskette.

```
disk/format –b /386/pbs –df /dev/fd0disk \
/386/9load /tmp/plan9.ini /386/9pcf.gz
```

Initialize the blank disk /dev/sdC0/data.

```
disk/mbr –m /386/mbr /dev/sdC0/data
disk/fdisk –baw /dev/sdC0/data
disk/prep –bw –a^(9fat nvram fossil cache swap) /dev/sdC0/plan9
disk/format –b /386/pbslba –d –r 2 /dev/sdC0/9fat \
/386/9load /386/9pcf /tmp/plan9.ini
```

FILES

/386/mbr

/386/mbr.bootmgr self–configuring `smart boot manager'

SOURCE

/sys/src/cmd/disk/prep

/sys/src/boot/pc

/n/sources/extra/bootmgr.tgz nasm assembler source for /386/mbr.bootmgr

SEE ALSO

floppy(3), sd(3), usb(4), 9boot(8), mk9660(8), mkusbboot(8), partfs(8)

BUGS

Format can create FAT12 and FAT16 file systems, but not FAT32 file systems. The boot block can only read from FAT12 and FAT16 file systems.

If prep –p doesn't find a Plan 9 partition table, it will emit commands to delete all extant partitions. Similarly, fdisk –p will delete all partitions, including data, if there are no partitions defined in the MBR.